CS 5500 HW3

Griffin Hackley A02224681

January 2021

After MPI initializes, process 0 becomes the cook and the other processes become the chefs.

Each chef will use an MPI probe to check to make sure that the cook is still in the kitchen. If he is not then the chefs leave as well. If he is still in the kitchen then they will wait between 1 and 5 seconds before sending an order.

The cook first checks to see if there are any orders with a MPI Probe. If there are no orders then he smokes a cigarette. If there are any orders then he gathers all of the orders up and returns an int that is the number of orders. If it is 21 or more then the cook quits and sends a kill message to the chefs. If it is not more than 20 orders then the cook will wait 1 second per order and then do it all over again.

The command to compile is: mpic++ hw3.cpp

The command to run is: mpirun -np numofprocesses a.out

The code is on the following two pages

Sample execution with one chef:

```
griffin@griffin-System-Product-Name:~/Desktop/HW3$ mpirun -np 2 a.out
Smoking a cig
Smoking a cig
Smoking a cig
Smoking a cig
Smoking a cig
Smoking a cig
got 1 orders
Smoking a cig
Smoking a cig
Smoking a cig
Smoking a cig
got 1 orders
Smoking a cig
got 1 orders
Smoking a cig
Smoking a cig
Smoking a cig
Smoking a cig
got 1 orders
Smoking a cig
Smoking a cig
Smoking a cig
^CSmoking a cig
```

Sample execution with 6 chefs:

```
griffin@griffin-System-Product-Name:~/Desktop/HW3$ mpirun -np 6 a.out
Smoking a cig
Smoking a cig
Smoking a cig
got 1 orders
got 2 orders
got 3 orders
got 4 orders
got 8 orders
got 7 orders
got 15 orders
got 13 orders
got 21 orders
Screw this, the cook has quit
The cook has quit, so chef 4 has quit
The cook has quit, so chef 2 has quit
The cook has quit, so chef 5 has quit
The cook has quit, so chef 3 has quit
The cook has quit, so chef 1 has quit
```

Cook Code:

```
int gatherOrders(){
    int flag, data;
    int count = 0;
    MPI_Status status;

    MPI_Iprobe(MPI_ANY_SOURCE, MPI_ANY_TAG, MCW, &flag, &status);

    while(flag){
        MPI_Recv(&data, 1, MPI_INT, MPI_ANY_SOURCE, 0, MCW, &status);
        count++;
        if(count > 20){
            break;
        }
        MPI_Iprobe(MPI_ANY_SOURCE, MPI_ANY_TAG, MCW, &flag, &status);
    }
    cout << "got " << count << " orders" << endl;
    return count;
}

void cook(int size){
    //initialize cook
    int data, count, orders = 0;
    int keepWorking = 1;
    int flag;
    MPI_Status status;

    while(keepWorking){
        //check for orders
        MPI_Iprobe(MPI_ANY_SOURCE, MPI_ANY_TAG, MCW, &flag, &status);

        //if there is at least one order, gather all of them
        if(flag){
            orders = gatherOrders();
        } else {
            //if no orders, cook smokes a cig
            cout << "Smoking a cig" << endl;
            sleep(1);
        }

        //if more than 20 orders, cook quits
        if(orders > 20){
            cout << "Screw this, the cook has quit" << endl;
            keepWorking = 0;
            data = -1;
                for(int i = 1; i < size; i++){
                MPI_Send(&data, 1, MPI_INT, i, 0, MCW);
            }
            break;
        }

        //go through orders
        while(orders > 0){
            orders--;
            sleep(1);
        }
    }
}
```

Chef Code:

```
void chef(int rank){
    //initialize chef
    int data, random, count = 0;
    int flag;
    int keepWorking =1;
    srand(time(NULL)*rank*rand());
    MPI_Status status;

    while(keepWorking){
        //make sure the cook hasnt quit
        MPI_Iprobe(0, -1, MCW, &flag, &status);
        if(flag){
            //if cook has quit, chef also quits
            MPI_Recv(&data, 1, MPI_INT, 0, 0, MCW, &status);
            if(data == -1){
                keepWorking = 0;
            }
        }

        //wait between 1 and 5 seconds
        random = (rand()%5) + 1;
        sleep(random);

        //send orders to cook
        // cout << "sent an order" << endl;
        MPI_Send(&data, 1, MPI_INT, 0, 0, MCW);
        count++;
    }
    cout << "The cook has quit, so chef " << rank << " has quit" << endl;
}
```